CSc 453 — Compilers and Systems Software

0 : Administrivia

Christian Collberg
Department of Computer Science
University of Arizona
`collberg@gmail.com`

August 24, 2009

1

Introduction

2 Contact Information

Class	:	453 — Compilers and Systems Software
Instructor	:	Christian Collberg
WWW	:	http://www.cs.arizona.edu/classes/cs372/fall08/index.html
Office	:	Gould-Simpson 758
Office Hours	:	MF 10:30-11:30
Phone	:	621-6612
Lectures	:	MF 9:00-10:15, F 9:00-9:50, GLDS 906
Honor's section	:	M 2:00-2:50, GLDS 942

3 Course Communication

Here are ways to communicate with me and the rest of the class:

- Email: `collberg@gmail.com`.
- Sign up for the class on `d2l.arizona.edu`.

4 Teaching Assistants

- Pankhuri
- Email: pankhuri@email.arizona.edu.
- Office: ?
- Office hours: Friday 4-5pm, Tuesday 4-5pm.

5 Exam-schedule

1. The midterm exam is scheduled for Fri Oct 16. This may change, so pay attention in class and check the web site.
2. The final exam is scheduled for Fri Dec 18, 8-10am.

6 Syllabus

You are responsible for reading and understanding this syllabus. If you have any concerns or issues about the information in this document you should bring them up during the first week of class.

7

Course Description

8 Topics

We will...

- learn how compilers are constructed,
- learn how programming languages are designed,
- learn how to compile procedural and object-oriented languages,
- learn how interpreters work,
- learn how garbage collectors work,
- learn how debuggers work (if there's time).

9 Syllabus

lexing

- The Chomsky hierarchy,
- regular expressions,
- DFAs,
- scanner implementation.

parsing

- Context-free grammars, BNF,
- parse trees,
- abstract syntax trees,
- Recursive Descent parsing.

10 Syllabus...

semantic analysis

- Attribute grammars,
- environments,
- type-checking.

intermediate representations

- stacks,
- tuples,
- trees,
- intermediate-code generation from abstract syntax trees.

11 Syllabus...

code generation

- control-flow graphs,
- code generation for arithmetic expressions, data-structure access, control-flow, and procedure calls.

code optimization

- survey of techniques,
- peephole optimization.

12 Syllabus...

- Advanced topics will be covered as time permits:

code optimization

- data-flow analysis,
- common sub-expression elimination,
- inlining.

object-orientation

- compiling Java-like languages,
- run-time class templates,
- inheritance.

13 Syllabus...

garbage collection

- reference counting,
- pointer-maps,
- mark-and-sweep.

systems software

- debuggers,
- profilers.

14

Course Work

15 Course Work (Subject to Change)

Assignment 1 Lexical analysis.

- Tokenize the Luca language.

  input: Luca program

  output: Lexical error messages, sequence of tokens.
- Algorithm: Finite state automaton.

16 Course Work (Subject to Change)

Assignment 2 Syntactic Analysis

- Parse a subset of the Luca language.

  input: Sequence of tokens.

  output: Syntactic error messages, AST.
- Algorithm: recursive descent.

Assignment 3 Semantic analysis.

- Perform type-checking for a subset of the Luca language.

  input: Luca AST

  output: Semantic error messages, decorated tree.
- Algorithm: recursive attribute evaluation.

17

Assignment 4 Interpretation, part I.

- Luca interpreter.

  input: Luca stack code

  output: none
- Implementation language: gcc.
- Algorithm: indirect threaded code.

Assignment 5 Interpretation, part II.

- Extend the interpreter from assignment 4 to handle procedures.

  input: Luca stack code

  output: none
- Algorithm: any.

18

Course Objectives

19 Course Objectives

- At the end of the course you should be familiar the overall design of compilers.
- You should know how to build lexers, parsers, semantic analyzers, code generators, interpreters, and runtime systems.

20 Course Methodology

- You can expect to spend a lot of time outside class on implementing a compiler.
- Each assignment builds on the previous ones — it's impossible to fall behind in this class and still pass it.
- The assignments are best done in pairs of two — you need to work on being a good collaborator and to organize your time and work appropriately.

21 Required extracurricular activities

- Programming assignments.
- Working programming exercises on your own.

22 Special materials required for the class

- None.

23 Assignment Format

- Assignments will be mostly in the form of programming problems.
- You may work the assignments on any machine you want, but before you hand them in *you should test the code on lectura!* The TA(s) will grade the assignments on lectura, and if they don't work there, he/she won't debug them for you! There can be subtle problems with code that's developed on a Windows machine, for example, when it is run on a Unix machine. For example, the two systems use different newline characters.

24 Prerequisites, Required Knowledge

- **Prerequisites:** C SC 345, C SC 352, C SC 372 recommended.
- You need to be a competent programmer in a procedural/object-oriented language, such as Java or C.

25

Assessment Scheme

26 Tests, Quizzes, and Assignments

There will be

1. one mid-term test worth a total of 10%;
2. one comprehensive final exam, worth a total of 40%;
3. five assignments worth a total of 50%;

27 Late Assignments

- Assignments handed in no more than 24 hours late will incur a 10% penalty.
- Assignments handed in more than 24 but no more than 48 hours late will incur a 20% penalty.
- Assignments handed more than 48 hours after the deadline will receive a grade of 0.

28 Making up Tests

You cannot make up the midterm or final exam unless

1. you have notified the instructor in writing (email is fine) or by phone prior to the test that you will be absent, and
2. you receive permission from the instructor to take the test at a later date.

29 Curving

- All grades (for exams assignments) will be curved up by throwing away the highest grade in the class and scaling up such that the second highest grade is 100.
- The curving is done to adjust for particularly difficult tests/assignments, and to prevent an outlier from skewing the grade distribution.
- You cannot, after scaling, receive more than 100 on any exam or assignment.

30 Grade Assignment

- You will fail the class if you get less than 50 (after curving) on the final exam.
- Otherwise, a curved total grade of [90,100] gives you an A, [80,89] a B, [70,79] a C, [60,69] a D, and 59 and below an E.

31 Incomplete work policy

- Except under exceptional circumstances I will not assign incomplete grades.
- I decide what is an exceptional circumstance.

32 Detailed Grading Scheme

- To avoid any ambiguities, I have formalized the informal rules given above.
- The rules below should be considered *minimum* requirements to achieve a particular grade. The instructor reserves the right to do additional adjustments, as necessary.
- Any contradictions, omissions, errors, or ambiguities in the grading scheme will be resolved by the instructor.
- Any issues or concerns regarding the grading scheme should be brought to the attention of the instructor within the first week of class.

33 Details — Curving

- All raw scores range from 0 to 100.
- Each individual score (final, midterm, assignments) will be curved using the function

  $$\mathrm{curve}(\bar{x}, s) = \min(100, (100.0/\max(\bar{x} - \max(\bar{x})))\bar{x}_s)$$

  where $\bar{x}$ is a set of scores (for an assignment, a test, etc.) and s is a student.
- Note: $-$ is set subtraction.
- $\mathrm{curve}(\bar{x}, s)$ returns s's score, curved up by 100.0/*2nd_highest_class_score*.

34 Details — Curving...

- For example, assume the following final exam scores:

$$34 \quad 45 \quad 66 \quad 88 \quad 98$$

  After the curve has been applied, the scores will be

$$38.6 \quad 51.1 \quad 75 \quad 100 \quad 100$$

35 Details — Exams

final exam:

- Let $\bar{f}$ be the set of final exam scores.
- Let $\bar{f}^s$ be the final exam score for student s.
- Let $\mathcal{W}^f$ be the weight of the final exam (40%).
- $\bar{t}_f^s = \text{curve}(\bar{f}, s)\mathcal{W}^f$ is the curved final score for s.

midterm exam:

- Let $\bar{m}$ be the set of midterm exam scores.
- Let $\bar{m}^s$ be the midterm exam score for student s.
- Let $\mathcal{W}^m$ be the weight of the midterm exam (10%).
- $\bar{t}_f^s = \text{curve}(\bar{m}, s)\mathcal{W}^m$ is the curved midterm score for s.

36 Details — Assignments

- Let $\bar{a}_i$ be the set of scores for the i:th assignment.
- Let $\bar{a}_i^s$ be the score for student s on the i:th assignment.
- Let $\mathcal{W}_i^a$ be the weight of the i:th assignment ($\sum_i \mathcal{W}_i^a = 50\%$).
- Let $\bar{\alpha}_i^s$ be the assignment score after late penalties have been applied:

$$\bar{\alpha}_i^s = \begin{cases} \bar{a}_i^s & \text{if the assignment is handed in on time} \\ 0.9\bar{a}_i^s & \text{if the assignment is} > 0 \text{ and} \leq 24 \text{ hours late} \\ 0.8\bar{a}_i^s & \text{if the assignment is} > 24 \text{ and} \leq 48 \text{ hours late} \\ 0 & \text{if the assignment is} > 48 \text{ hours late} \end{cases}$$

37 Details — Assignments...

- $\bar{t}_a^s = \sum_i(\text{curve}(\bar{\alpha}_i, s)\mathcal{W}_i^a)$ is the total curved assignment score for student s.
- If, for whatever reason, the actual number of assignments is less than the planned number, the $\mathcal{W}_i^a$'s will be scaled up uniformly.

38 Details — Total Scores

- The raw total score for student s is

$$\bar{t}_s = \bar{t}_f^s + \bar{t}_m^s + \bar{t}_a^s$$

- We round up to the nearest integer:

$$\text{total}_s = \lceil \bar{t}_s \rceil$$

39 Details — Grade Assignment

- The final grade assignment for student s is

$$\text{grade}_s = \begin{cases} E & \text{if } t_f^s < 50 \\ \begin{cases} A & \text{if total}_s \in [90, 100] \\ B & \text{if total}_s \in [80, 89] \\ C & \text{if total}_s \in [70, 79] \\ D & \text{if total}_s \in [60, 69] \\ E & \text{if total}_s < 60 \end{cases} & \text{otherwise} \end{cases}$$

- In other words, a student with a curved final exam score $t_f^s < 50$ will fail the class, regardless of their results on the other assessment categories.

40

Policies

41 Office hours

- Office hours: MF 10:30-11:30.
- In addition to the office hours I use an open door policy:

42 Collberg's Café

- Please come and see me to chat, ask questions, or snack:

43 Attendance Policy

- My goal is to keep class attendance high so that we can get good discussions going in the class.
- You are not required to attend lectures, but...

  you cut class at your own risk.

  Anything covered in class or in any of the required readings is fair game on tests and exams.

44 Attendance Policy...

- All holidays or special events observed by organized religions will be honored for those students who show affiliation with that particular religion. Absences pre-approved by the UA Dean of Students (or Dean's designee) will be honored.

45 Subject to Change Policy

- The information contained in this course syllabus, other than the grade and absence policies, may be subject to change with reasonable advance notice, as deemed appropriate by the instructor.
- The instructor reserves the right to
  1. add, drop, or change topics;
  2. change exam or homework dates, etc.
- *Changes will be announced in class and on the class web site! You are responsible for checking this site regularly. Also check `d2l.arizona.edu`.*

46 Notification of Objectionable Materials

- There is no objectionable material in this class.

47 Handicapped Accessibility

Students with disabilities who require reasonable accommodations to fully participate in course activities or meet course requirements must register with the Disability Resource Center. If you qualify for services through DRC, bring your letter of accommodations to me as soon as possible. See `http://www.salt.arizona.edu/`.

48 Student Code of Academic Integrity

- Assignments in this course require individual attention and effort to be of any benefit. All work is expected to be that of each student alone. You may not consult with others, except in ways specifically authorized by the course instructor. You also may not plagiarize another person's work or copy another person's code.

49 Student Code of Academic Integrity...

- Students are responsible for understanding and complying with the University's Code of Academic Integrity. A synopsis of the Code is attached; the full text is available from the Office of the Dean of Students in Room 203 Old Main. Among other provisions, the Code demands that the work you submit is your own, and that graded papers and exams will not subsequently be tampered with. Copying of another student's programs or data, or writings is prohibited when they are part of a published class assignment; it is immaterial whether the copying is by computer, xerox, pen or other means. Witting collaboration in allowing such copying is also a Code violation.

50 Student Code of Academic Integrity...

- Assignments in this course require individual attention and effort
- Violations of the Code will, at minimum, result in loss of credit for a graded item. An egregious first violation or any second violation will minimally result in failure of the entire course.
- See also `http://studpubs.web.arizona.edu/policies/cacaint.htm` the University of Arizona Code of Academic Integrity.

I take academic integrity seriously! I will report every *violation!*

51 Expected classroom behavior

- Be courteous and treat others in the class with respect.
- Please be courteous to other students by refraining from talking, playing loud music in your headphones, silencing cell phones, pagers, etc.
- We come to class to learn: don't read the newspaper, solve cross-word puzzles, etc.
- Treat the TAs with respect: they do their best to grade your assignments on time, help you with software installation problems, help you with assignments, etc. But they have their own class work to attend to, too.

52 Policies against threatening behavior

- Read and abide by the following link: `http://policy.web.arizona.edu/~policy/threaten.shtml`.

53

Teaching Material

54 Handouts & Other Material

1. The textbook is Louden – "Compiler Construction: Principles & Practices". Other books (Appel – "Modern Compiler Implementation in Java.", Aho, Sethi, Ullman – "Compilers – Principles, Techniques, and Tools" (*The Dragon Book*)) will do equally well.
2. I always make copies of my transparencies available to students. Note that
   - I do this to relieve you of having to take notes during lectures,
   - they are not substitutes for reading the textbook,
   - their primary purpose is to remind you of what you need to study for the exam.

55

Why study compilers?

56 Why Am I a Compiler Jock?

From: `Dwight VandenBerghe, dwight@pentasoft.com`

Newsgroups: `comp.compilers`

```
I was seventeen and had a lot of time on my hands.  I was a programmer for the US Marines,
stationed in Da Nang, Vietnam, in the mid sixties, and during the long nights I would go through
the IBM microfiche - they kept a full set in the computer room, all the manuals and all the
source code for all their tools.  I got interested in the COBOL compiler, and I read through
the assembler code for it.  I still remember reading the instructions that scanned in an integer.
It just thrilled me, seeing how that magic was done.
```

57

```
I stayed up late into the early morning, trying to figure it all out.
   It's 3:10AM here now, and I've just finished downloading some papers from Norway on attribute
grammars in ML. I have to go print them out.  I'm dead tired, but this stuff is so exciting
that sleep will have to wait.  I'm 48 now, and my youngest boy is almost seventeen.  My grandkids
are great, and my wife is a gem, but you know, I hope that I am still cheating sleep for another
couple of decades, because I feel seventeen, not nearly fifty.  Compilers are magical.  Compilers
are like the two opposing mirrors in the funhouse - you see infinity, yet you are clearly finite.
```

58

Writing a compiler, you go back and forth, from the infinite to the practical, over and over again.

And now, with the advent of functional programming into my life, there is an elegance to the code that I haven't seen before. The functional style seems to me to be made for us compiler jockeys.

And the advances that have been made in this field: BURS theory, the SUIF system, transformational tree rewrite systems, higher-order attribute grammars, the exciting new work with LL(infinite) parser generators ... it's simply thrilling. Nothing else compares, at least, for me. Applications suck; device drivers are tedious; operating systems and file management and databases are boring.

59

After 31 years, I've pretty much done it all, and if it wasn't for compilers I'd be bored stiff. Yet here it is, the wee hours yet again. Last night I finally hauled myself to sleep at a little after 2AM, because I found some great papers on polymorphic type theory on some server in the UK. So it goes.

Dwight

60 What's available in Compiler Jobs?

From: Clifford Click, cliff.click@Eng.Sun.COM

Newsgroups: comp.compilers

> 2. What do you think is the future for this market?
> 3. Is the demand growing?

Skilled *optimizing* compiler writers are in high demand. The market is small, but the the labor pool is even smaller. (The demand for people who make functioning parsers is much smaller, probably due to the ease of using lex & yacc).

61

The future for this market? Welll... as long as chip architects and language designers are alive I got a job. Merced looks like full time employment for 20+ people for 5+ years (OUTSIDE of Intel & HP; they probably have double that number already and are always looking for more). [Again, I'm talking about optimizing compiler writers; big optimizers grow big support groups that hire more people in nearby fields]

And yes, I believe the demand is growing.

62 www.compilerjobs.com/jobs/microsoft.htm

```
Job Title      Software Design Engineer Lead
Description    The Visual Basic.Net team is in search of an
               exceptionally strong development lead,
               experienced and interested in driving the
               development effort on Visual Basic .Net
               compiler. Responsibilities include leading
```

	a team in designing and implementing new VB.Net language features. This is a great opportunity to be involved with the development of the most successful development tool and one of the most important technologies at Microsoft.
Qualifications	Solid knowledge of C/C++ and multi-threaded programming. A BA/BS or MS degree in CS.
Website	www.microsoft.com

63 www.compilerjobs.com/jobs/cray.htm

Job Location	Mendota Heights, MN USA
Description	Cray Inc. designs, builds, and sells high-performance MPP, vector processor and scalable shared memory parallel computer systems.
Qualifications	Proficient in C, C++, and Fortran. Minimum of 2-3 years experience in writing compilers. Familiar with Unix. Experience is desired in: Compiling for Multiprocessors, Parallel programming, Performance tuning.
Education:	B.S. in Computer Science or equivalent experience.
Our website	http://www.cray.com

64 www.compilerjobs.com/jobs/cloakware.htm

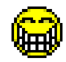
Job Title	Compiler Designer - Permanent
Job Location	Ottawa, Ontario, CANADA
Description	Working with our core technology team you will develop compilers for our Tamper-Resistent Software (TRS).
Our website	http://www.cloakware.com
Qualifications	Cloakware is seeking expert Compiler Writers, with knowledge any or a combination of the following: Optimization, Program Transformation, Combinatorics, Obfuscation Techniques, Program slicing Techniques, Numerical Analysis, Semetric Representations for Control- and Data-Flow, Control- and Data-Flow Dependencies

65 www.compilerjobs.com/jobs/irvine.htm

Job Title	Post-Doctoral Researcher
Description	Several postdoctoral research positions are available, working on new approaches to mobile code; this is compiler-related research.
Qualifications	Applicants must have attained a Ph.D. in Computer Science or related field and have prior experience with software systems, possess demonstrated familiarity with code generation for modern RISC architectures, and the Ada and Java programming languages.

Website www.ics.uci.edu/~franz

66 Now What?

Let's Have Fun!!![1] 

[1] That's right — compilers are fun!